

Bare metal K8s,
или Туда и обратно

История Quadcode

Устинов Илья
Kubernetes Platform

HL
HighLoad++
2022

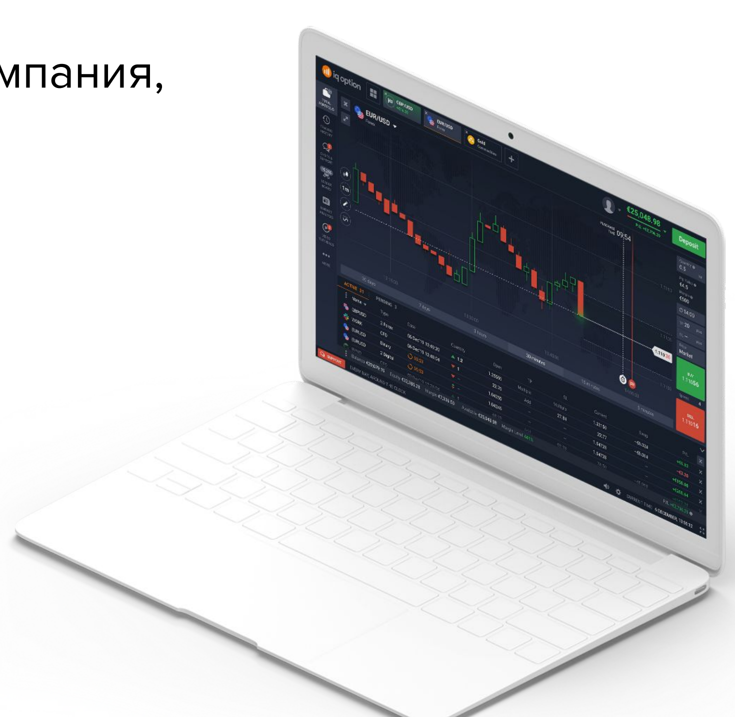
quadcode

1

Про компанию



Quadcode — международная продуктовая компания, которая создаёт программное обеспечение для торговой и инвестиционной индустрии.





Пользователей

17

Языков



Установок приложений

218+

Стран

Моя команда





Юра
Team Lead



Игорь
DevOps



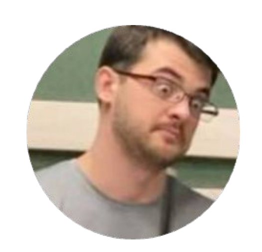
Кирилл
DevOps



Рома
DevOps



Антон
Product Manager



Илья
DevOps



Костя
Product Owner

И принимавшие участие ранее



Вова
DevOps



Саша
Go Developer



Денис
DevOps



Сергей
DevOps



Даша
Dev



Вова
DevOps



Паша
Dev



Миша
DevOps



Стас
DevOps

Что будет в докладе



Ретроспектива жизни Bare Metal-кластера, а также*:

- Как докатиться до 4 часов при скейле ноды.
- Как сократить это время до 3 минут, когда уже докатились.
- Как AWS, GCE, DO собирают свои кластеры, и что можно перенять у них в Bare Metal.

* возможны флешбэки

Про наши кластеры



3 k8s-кластера, по одному в окружениях:
Infra, Preprod, Prod

	Prod	**Preprod**	**Infra**
Количество нод	97 nodes	53 nodes	22 nodes
Каждая нода	40 CPU/256 GB	40 CPU/256 GB	40 CPU/256 GB
Общая ёмкость	3880 CPU, 24.25 TB	2120 CPU, 13.25 TB	880 CPU, 5.5 TB

Предпосылки.
До 2017 года

Предпосылки





Предпосылки. Архитектура



Монолит → Микросервисы

Monolith
Presentation
Business
Persistence

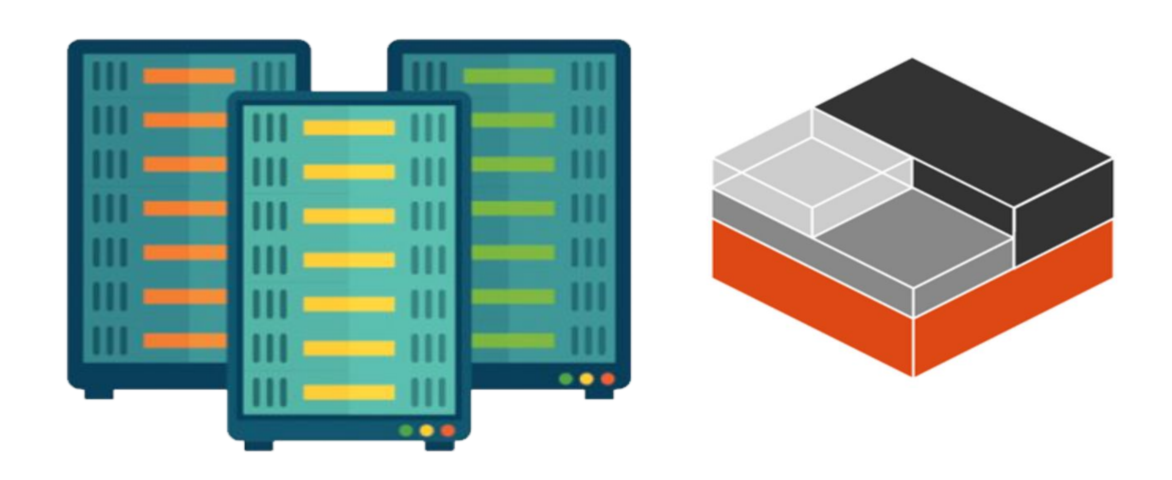
Microservices
Service1
Service2
Service3
Service4



Предпосылки. Архитектура



Bare Metal + LXC Containers



Предпосылки. Архитектура



Отдельная команда эксплуатации. Процесс заказа серверов.

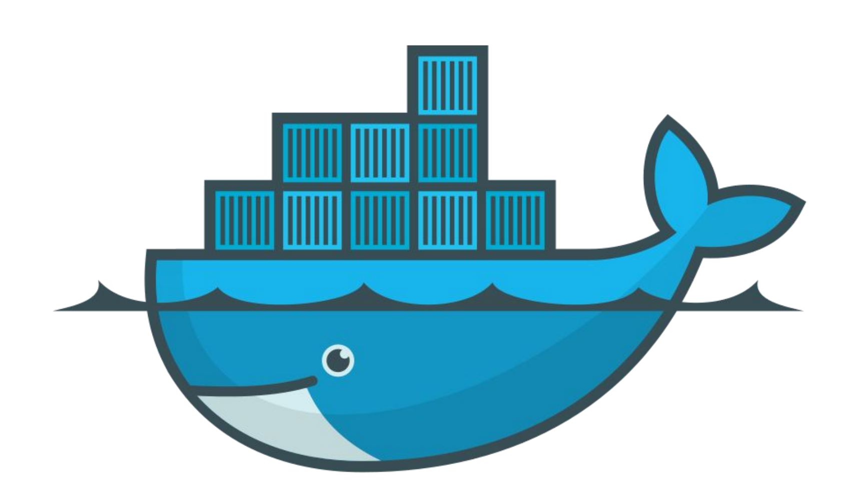
мы настроили

Сервер → Infra-support шебуршит начальную настройку → K8S Platform / OPS / OpenStack

Предпосылки. Архитектура



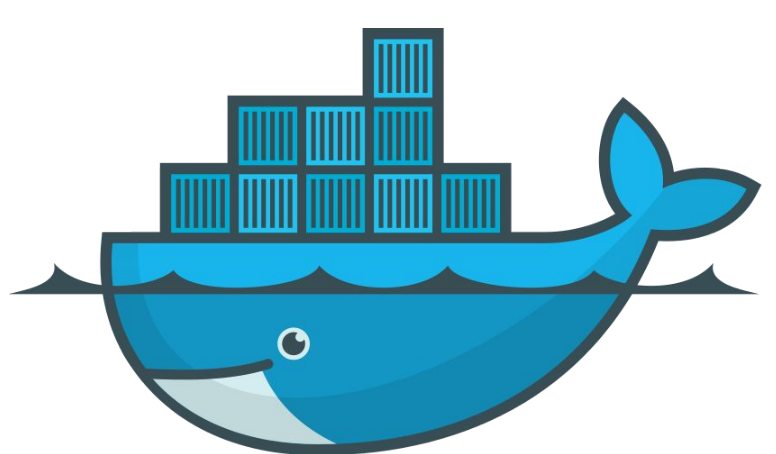
Docker для dev- и test-окружений



Предпосылки. Итого



- Модное на тот момент желание переехать на микросервисы.
- Много разного неудобного рантайма: вроде LXC, вроде где-то просто настроенные сервера.

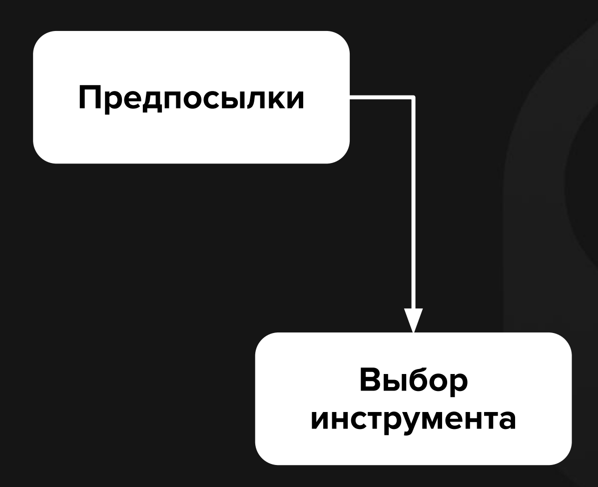




Хотим один рантайм, чтобы править всеми!



Выбор инструмента оркестрации. Q2 2017



Выбор инструмента. Self-hosted или облако

quadcode

Self-hosted:

- Можно потрогать.
- Требования безопасности и бизнеса:
    - Всё должно быть во внутреннем сетевом периметре.
    - Низкая latency.

Облако:

- Нельзя потрогать.
- Ходят слухи, что управлять облаком проще.
- Требования безопасности и бизнеса:
    - Всё должно быть во внутреннем сетевом периметре — нужно идти к noc'ам и узнавать, сколько стоит.
    - Низкая latency — нужно идти к noc'ам.

Выбор инструмента.
Ожидаемая стоимость кластера, просто



Железо:

- Мастера и рефлекторы поменьше 8 CPU/64 GB — €190 в месяц × 6.
- Ноды побольше 40 CPU/256 GB — стандарт поставки дата-центра €585 в месяц × 5 для начала.

€55 800 в год

Облако:

- Amazon EKS — €75 в месяц.
- Amazon EC2 — €6603 в месяц.

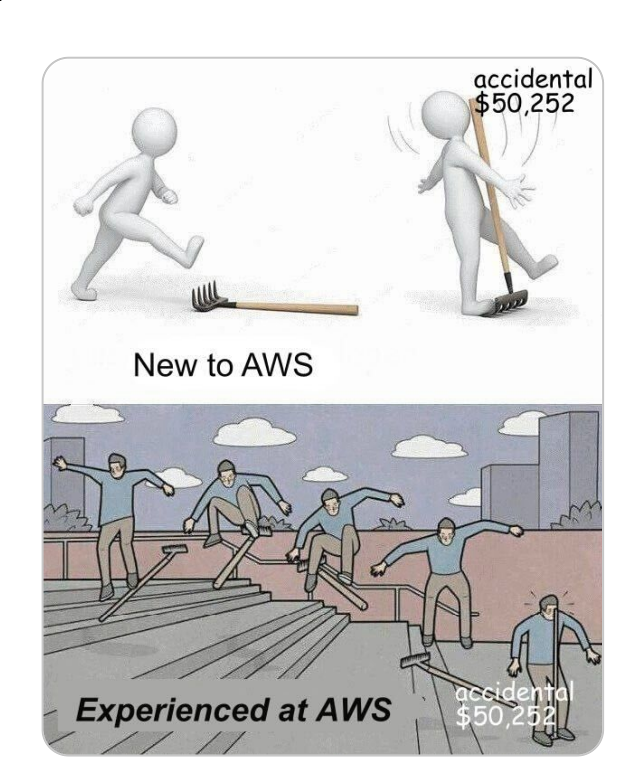
€78 132 в год

Выбор инструмента.
Ожидаемая стоимость кластера



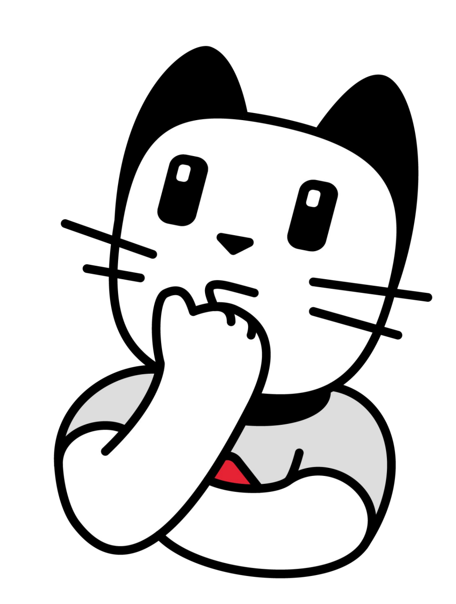
Дополнительная инфраструктура облака —
€78 132 плюс стоимость:

- VPC.
- TGW.
- Direct connect.
- ALB, ELB, NLB, NLP.
- VPC peering.
- ETC.



Выбор инструмента.
Ожидаемая стоимость кластера, сложно.
Экспертиза, люди, процессы





Выбор инструмента. Решения для Bare Metal



Если bare metal, то какой?



- Поднимает кластер в облаке на EC2







Kubernetes The Hard Way by Kelsey Hightower

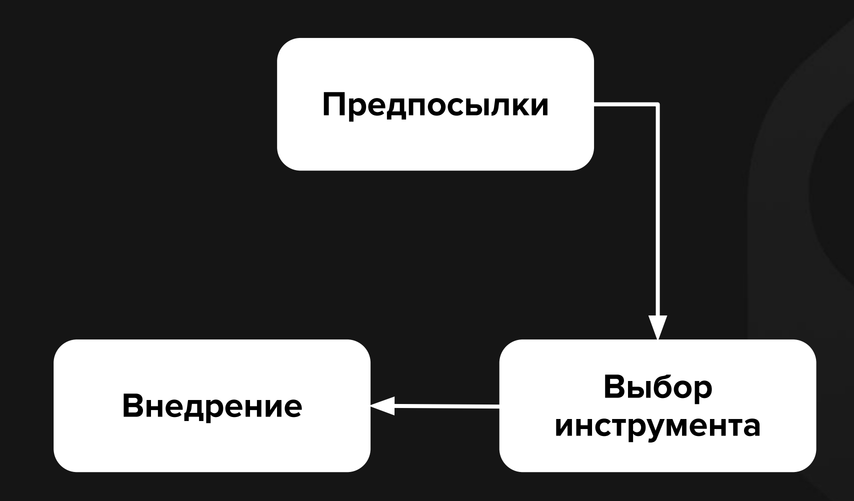
- Очень подробный
- Идеально, чтобы учиться
- Нужно автоматизировать

Выбор инструмента. Kubespray:

quadcode

- Написан на Ansible.
- Позволяет настраивать компоненты.
- Подготавливает окружение.
- Разрабатывался Mirantis.

Внедрение.
Q3 2017



Внедрение.
Что включила в себя эксплуатация

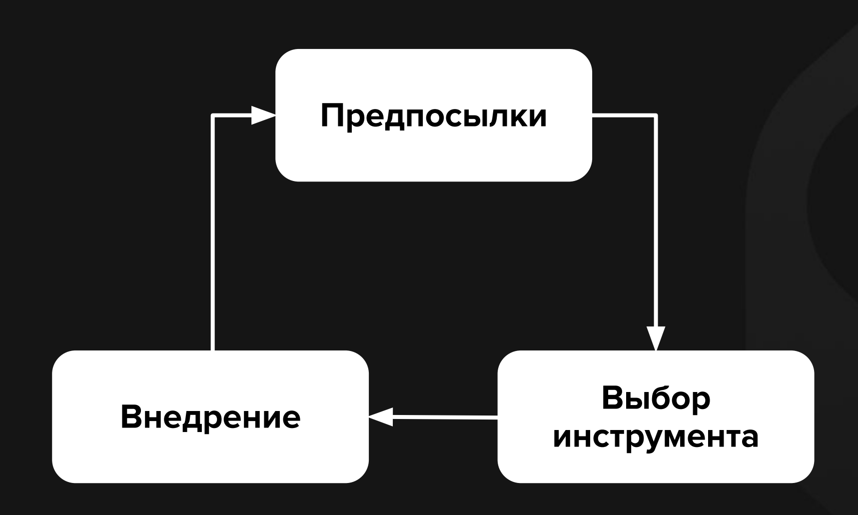


Стандартные операции:

- Раскатка кластера.
- Скейл кластера.
- Разработка процесса для ввода приложений в k8s, авторизации, рекомендуемых пайплайнов деплоя, etc.
- Апгрейд софта ноды: Docker, Kernel, etc.
- Апгрейд версии кластера.
- Обновления сертификатов k8s, CNI, etc.

Размер кластера:
5 → 20 нод

Предпосылки рефакторинга. 2017-2019





Перерыв на cool story



Предпосылки. Проблемы



Физический переезд Master-серверов

Старый DC

Новый DC

Предпосылки. Рост времени выполнения стандартных операций



Стандартные операции:

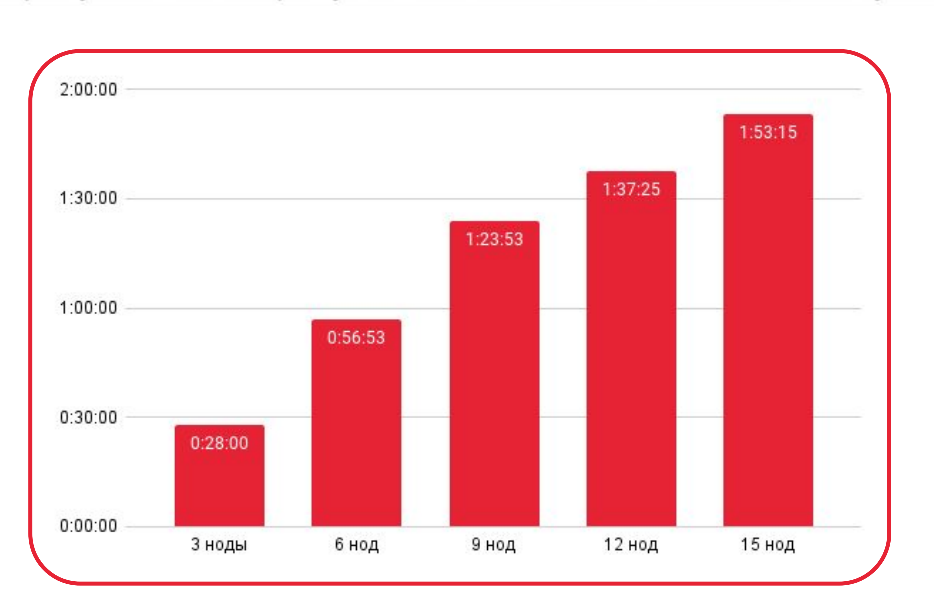
- Раскатка кластера: 0,5 часа → 4 часа.
- Скейл кластера: 0,5 часа → 4 часа.
- Разработка процесса для ввода приложений в k8s, авторизации, рекомендуемых пайплайнов деплоя, etc.: N спринтов → поддержка.
- Апгрейд софта ноды (Docker, Kernel, etc.): несколько дней.
- Апгрейд версии кластера: не пробовали → квартал с двумя кластерами.

Предпосылки. Рост времени выполнения стандартных операций



Сейчас время выполнения Kubespray в sample-конфигурации:

```
root@b26f77349123:/kubespray# ansible-playbook -u ubuntu -b -i inventory/inventory.ini cluster.yml
```



Предпосылки. Чему мы научились и какие выводы сделали



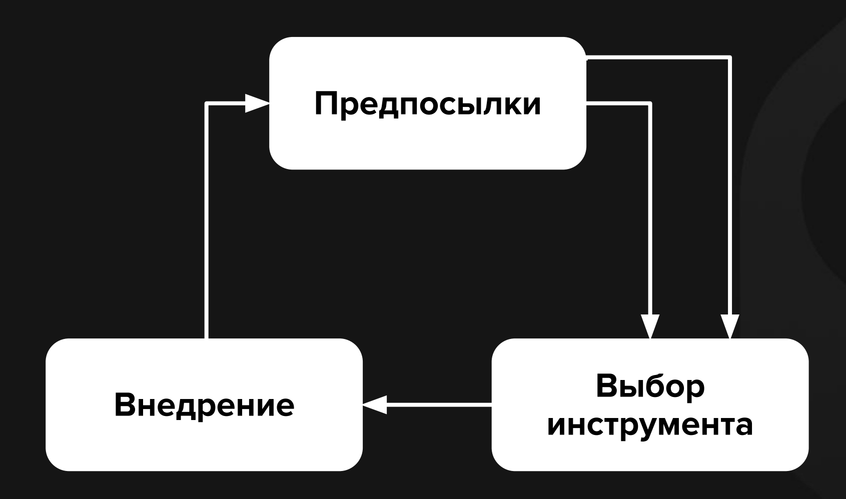
Разобрались:

- В переменных Kubespray.
- Как работает CNI.
- Как работает k8s.

Какие выводы:

- Kubespray долго скипает таски.
- Нужен не любой, а наш, настроенный, работающий кластер с нашими переменными.
- 3 года назад мастера через Kubespray скейлить нельзя, но мы примерно представляем как.
- Нужен инструмент, которому можно делегировать стандартные операции.

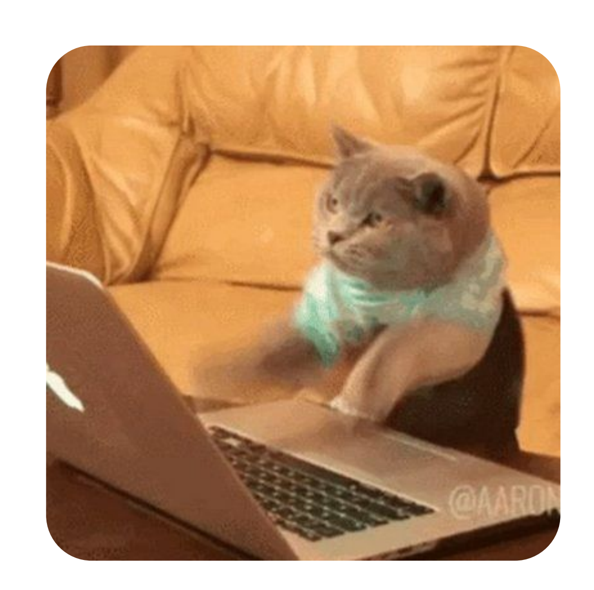
Выбор инструмента
Q3 2019



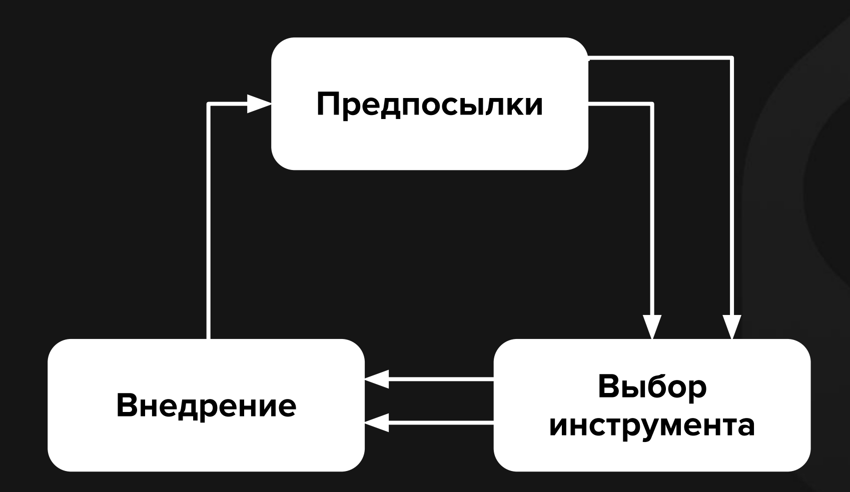
Выбор инструмента. Kubeadm



- Вышел из беты.
- Команда получила контекст, поняли, что можем делегировать некоторые процессы.
- Kubeadm становится рекомендуемым способом оперирования k8s-кластером, в том числе и в официальной документации.



Внедрение.
Q4 2019



Внедрение. Свой плейбук

quadcode

```yaml
# Create audit policy files on each master (for apiserver and falco)
- hosts: kube-master
  become: yes
  roles:
    - { role: kubernetes-audit-policy, tags: "kubernetes-audit-policy" }
  tags: ["setup-cluster", "k8s-audit-policy"]

# Setup first master (kubeadm init). It is executed only on first play on firs master.
- hosts: kube-master[0]
  become: yes
  roles:
    - { role: kubeadm-init, tags: "kubeadm-init" }
  tags: ["setup-cluster", "kubeadm-init"]

# Generate join tokens and join new masters/nodes in cluster.
- hosts: kube-master[0]
  become: yes
  roles:
    - { role: kubeadm-join, tags: "kubeadm-join"}
  tags: ["setup-cluster", "kubeadm-join"]

# Setup calico using helm
- hosts: kube-master[0]
  become: yes
  roles:
    - { role: kubernetes-networking, tags: "kubernetes-networking"}
  tags: ["setup-cluster", "kubernetes-networking"]

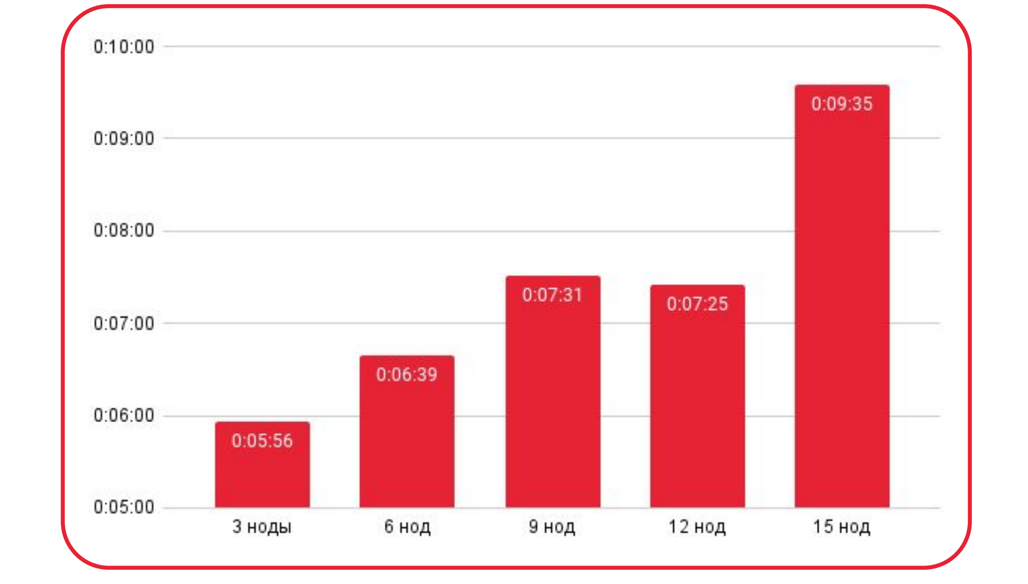
# Add labels and anotations on nodes for manage taints using helm:
# app.kubernetes.io/managed-by: Helm
# meta.helm.sh/release-name: node-taints-labels
# meta.helm.sh/release-namespace: kube-system
- hosts: kube-master[0]
  become: yes
```

Что включила в себя эксплуатация



Стандартные операции:

- Раскатка кластера.
- Скейл кластера.
- Апгрейд софта ноды: Docker, Kernel, etc.
- Апгрейд версии кластера.
- Обновления сертификатов k8s, CNI, etc.



Размер кластера:

20 → 100 нод

Что мы поняли еще через 2 года

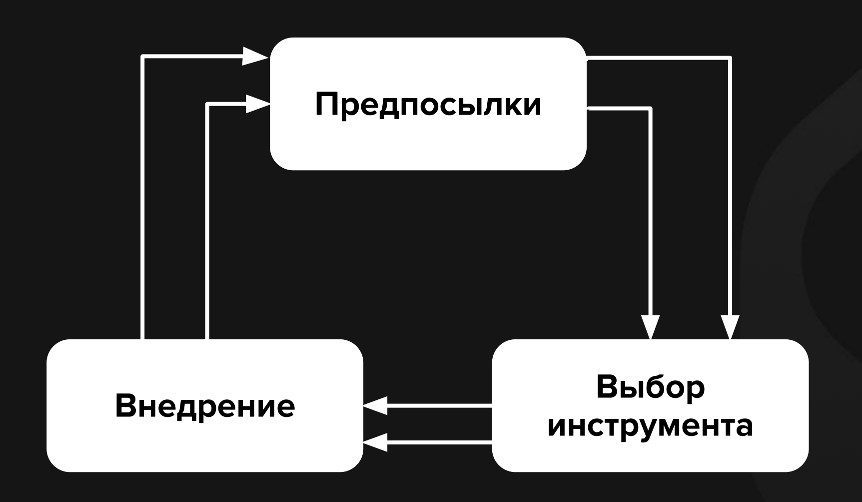


Стандартные операции:

- Раскатка кластера: 10 минут → 1,5 часа.
- Скейл кластера: 10 минут → 1,5 часа.
- Апгрейд софта ноды (Docker, Kernel, etc) → несколько дней.
- Апгрейд версии кластера: пробовали → Kubeadm стал поддерживать апгрейд с 1.17.

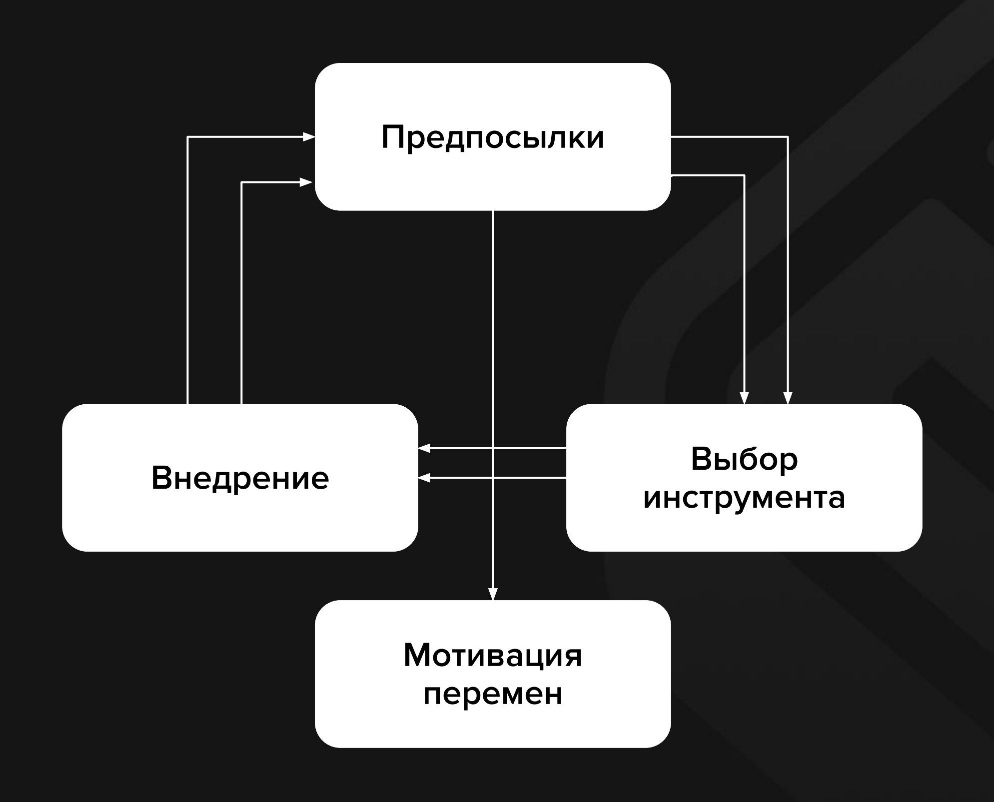
Пришлось переезжать ещё раз в течение полугода.

Кажется, мы всё это проходили Q3 2021





Мотивация


Предпосылки
Внедрение
Выбор инструмента
Мотивация перемен

Мотивация. Та же самая задача — развернуть кластер, но:



- Время поднятия < 3 минут на EC2.
- Число нод — произвольное.
- Число кластеров — произвольное.

Текущий флоу не подойдёт, потому что:

- Неудобно каждый раз прогонять плейбук на все тестовые среды, которых у нас ~200 в день.
- Мы не сможем сделать это руками даже с запуска джоб.
- Чтобы делать динамический инвентори, нужно делать автоматизацию.
- Если нужно делать автоматизацию, то зачем автоматизировать прокатку плейбука, если нам нужно автоматизировать поднятие кластера.

Мотивация. Profits



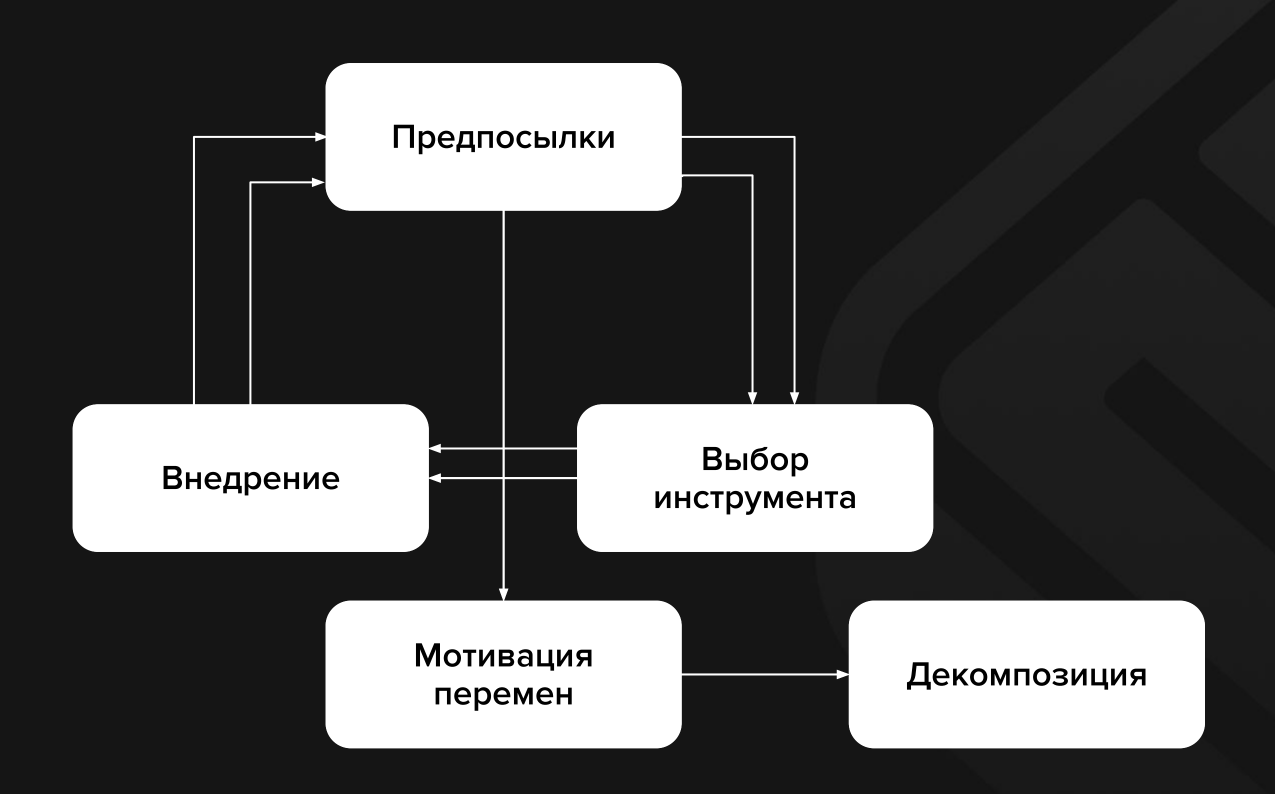
- Можно обеспечить постоянную и ежедневную проверку конфигурации на работающих тест-окружениях.
- Где-то здесь должно быть что-то, чтобы не катать наши железки 4 часа, попивая чай и ожидая stderr /dev/null от ансибла на одной из 100 нод.

```
fatal: [illegible] : FAILED! => {}

MSG:

Failed to get information on remote file (/root/.bashrc): Couldn't open /dev/null: Operation not permitted
```

Декомпозиция процесса



Декомпозиция процесса. Hard way



Вернёмся в самые дебри и посмотрим, что предлагает hard way:

Этапы по hard way	**Число команд**
Провижн CA и создание сертификатов для control plane	31
Генерация конфигов k8s для аутентификации	21
Генерация конфига и ключа шифрования	3
Бутстрап etcd	13 на каждом члене etcd-кластера
Бутстрап k8s control plane	33
Бутстрап нод	25
Конфигурация kubectl	5
Установка CNI	3
Установка DNS	6

Декомпозиция процесса. Документация k8s



Kubeadm

- Провижн CA и создание сертификатов для control plane
- Генерация конфигов k8s для аутентификации
- Генерация конфига и ключа шифрования
- Бустрап etcd
- Бустрап k8s control plane

Kubeadm init —config $bestdir/bestkubeadm.conf

Бустрап etcd
Бустрап k8s control plane

```
kubeadm join
k8s-apiserver.build1267216443
--token youcanusemeanywhere
--discovery-token-ca-cert-hash
sha256:trytodecryptmehaha
--control-plane --certificate-key
forthebestbackdooryoeverfound
```

Бустрап нод

```
kubeadm join
k8s-apiserver.build1267216443
--token youcanusemeanywhere
--discovery-token-ca-cert-hash
sha256:trytodecryptmehaha
--certificate-key
forthebestbackdooryoeverfound
```

Установка CNI

```
helm install calico
projectcalico/tigera-operator
--version v3.23.1 --atomic
--namespace tigera-operator
--kubeconfig=/root/.kube/config
```

Декомпозиция процесса. Lifehack



А ноду забутстрапить можно и просто вот так:

Заранее сгененируйте для kubelet'а сертификаты для авторизации

```
client-certificate: /var/lib/kubelet/pki/kubelet-client-current.pem
client-key: /var/lib/kubelet/pki/kubelet-client-current.pem
```

Доставьте на ноду вместе с

```
"clientCAFile": "/etc/kubernetes/pki/ca.crt"

/etc/kubernetes/kubelet/kubelet-config.json
```

И запустите kubelet с ключом --register-node

Прототипирование



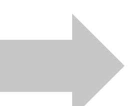
Прототипирование. Делаем прототип



Ставим плейсхолдеры в groups vars, для kubeadm.conf

```yaml
dex_endpoint: 'https://--DEX_DOMAIN--/'
kube_apiserver_fqdn: '--API_SERVER_DOMAIN--'
etcd_endpoints: ['--ETCD_DOMAIN--']

kubernetes_version: 1.21.13
kubernetes_utils_version: "{{ kubernetes_version }}-00"
```



```
root@localhost:/etc/kubernetes# cat kubeadm.conf
apiVersion: kubeadm.k8s.io/v1beta2
kind: ClusterConfiguration
imageRepository: k8s.gcr.io
certificatesDir: /etc/kubernetes/pki
clusterName: kubernetes
dns:
  type: CoreDNS
kubernetesVersion: "v1.21.13"
controlPlaneEndpoint: --API_SERVER_DOMAIN--:6443

etcd:
  local:
    serverCertSANs:
    - --ETCD_DOMAIN--
    peerCertSANs:
    - --ETCD_DOMAIN--
    dataDir: /var/lib/etcd
```

Прототипирование. Делаем прототип



Прописываем метаданные инстансам:

```
~ cat ud-decoded
{
 "domainsEnv": "BASE_DOMAIN=*
API_SERVER_DOMAIN=* ETCD_DOMAIN=*",
 "masterAddr": "master01.build126.*",
 "kubenodesAddr": "node01.build126.*
node02.build126.* node03.build126.*",
 "myDomain": "node03.build126.*",
 "myRole": "node"
}
```

Current user data

User data currently associated with this instance

"eyJkb21haW5zRW52IjoiUWtGVFJWOUVUMDFCU1U0OUltSjFhV3hrTVRJMk56RXpMbk5pTG5GalpYZ3VZMjl0SWdw
QlVFbGZVMFZTVmtWU1gwUlBUVUZKVGowaWF6aHpMV0Z3YVhObGNuWmxjaTVpZFdsc1pERXlOamN4TXk1ellpNX
hZMlY0TG1OdmJTSUtSVlJEUkY5RVQwMUJTVTQ5SW10MVltVnRZWE4wWlhJd01TNWlkV2xzWkRFeU5qY3hNeTV6
WWk1eFkyVjRMbU52YlNJS1JFVllYMFJQVFVGSlRqMGlhemh6TFdSbGVDNWlkV2xzWkRFeU5qY3hNeTV6WWk1eFkyV
jRMbU52YlNJS1RrOUVSVk5mUkU5TlFVbE9VejBpYTNWaVpXMWhjM1JsY2pBeExtSjFhV3hrTVRJMk56RXpMbk5pTG5
GalpYZ3VZMjl0SUd0MVltVnViMlJsTURFdVluVnBiR1F4TWpZM01UTXVjMkl1Y1dObGVDNWpiMjBnYTNWaVpXNXZaR
1V3TWk1aWRXbHNaREV5TmpjeE15NXpZaTV4WTJWNExtTnZiU0JyZFdKbGJtOWtaVEF6TG1KMWFXeGtNVEkyTnpFe
kxuTmlMbkZqWlhndVkyOXRJR3QxWW1WdWIyUmxNRFF1WW5WcGJHUXhNalkzTVRNdWMySXVjV05sZUM1amIyMG
dhM1ZpWlc1dlpHVXdOUzVpZFdsc1pERXlOamN4TXk1ellpNXhZMlY0TG1OdmJTQnJkV0psYm05a1pUQTJMbUoxYVd
4a01USTJOekV6TG5OaUxuRmpaWGd1WTI5dElnPT0iLCAibWFzdGVyQWRkciI6ImEzVmlaVzFoYzNSbGNqQXhMbUoxY
Vd4a01USTJOekV6TG5OaUxuRmpaWGd1WTI5dCIsICJrdWJlbm9kZXNBZGRyIjoiYTNWaVpXNXZaR1V3TVM1aWRXbH
NaREV5TmpjeE15NXpZaTV4WTJWNExtTnZiU0JyZFdKbGJtOWtaVEF5TG1KMWFXeGtNVEkyTnpFekxuTmlMbkZqWlhn
dVkyOXRJR3QxWW1WdWIyUmxNRE11WW5WcGJHUXhNalkzTVRNdWMySXVjV05sZUM1amIyMGdhM1ZpWlc1dlpHVXdOQ
VXdOQzVpZFdsc1pERXlOamN4TXk1ellpNXhZMlY0TG1OdmJTQnJkV0psYm05a1pUQTFMbUoxYVd4a01USTJOekV6TG5
G5OaUxuRmpaWGd1WTI5dElHdDFZbVZ1YjJSbE1EWXVZblZwbGZwYkdReE1qWTNNVE11YzJJdWNXTmxlQzVqYjIwaWSIsICJt
eURvbWFpbiI6Imt1YmVub2RlMDMuYnVpbGQxMjY3MTMuc2IucWNleC5jb20iLCAibXlSb2xlIjoibm9kZSJ9"

Прототипирование. Делаем прототип



Пишем скрипт в cloud-init, чтобы забрать метаданные и запустить поднятие кластера:

```
function main () {
  log w "${FUNCNAME[0]}" "Starting Kubernetes cluster setup or join to existed"
  downloadMetadata
  extractMetadata
  exportGenaralUsageVars
  setHostname "${HOSTNAME}"
  case "${ROLE}" in
    "master" )
      setupMaster
    ;;

    "masterFollower" )
      setupMasterFollower
    ;;

    "node" )
      setupNode
    ;;

    * )
      log e "${FUNCNAME[0]}" "No instructions for role ${ROLE}"
      exit 1
  esac
}
```

Прототипирование. Что получилось



- Правда 3 минуты.
- Правда сколько угодно кластеров, пока есть инстансы в облаке.
- Правда непонятно, как скейлить и обновлять, но об этом позже. Мы же тут для тестовых коротко живущих сред поднимаем (ну, в общем случае).

Прототипирование. Как у вендоров?



EKS (join nodes)

```
root@ip-10-236-98-29 /]# grep eks /var/log/cloud-init-output.log
 API_SERVER_URL=https://58F9FA156AC4C6840A53C91EA04A9AB9.gr7.eu-central-1.eks.amazonaws.com
 /etc/eks/bootstrap.sh delete-me --kubelet-extra-args '--node-labels=eks.amazonaws.com/nodegroup-image=ami-004e93
```

```
TOKEN=$(get_token)
AWS_DEFAULT_REGION=$(get_meta_data 'latest/dynamic/instance-identity/document' | jq .region -r)
AWS_SERVICES_DOMAIN=$(get_meta_data 'latest/meta-data/services/domain')
MACHINE=$(uname -m)
PAUSE_CONTAINER_ACCOUNT=$(get_pause_container_account_for_region "${AWS_DEFAULT_REGION}")
PAUSE_CONTAINER_IMAGE=${PAUSE_CONTAINER_IMAGE:-$PAUSE_CONTAINER_ACCOUNT.dkr.ecr.$AWS_DEFAULT_REGION.$AWS_SERVICES_DOMAIN/eks/pause}
PAUSE_CONTAINER="$PAUSE_CONTAINER_IMAGE:$PAUSE_CONTAINER_VERSION"
CA_CERTIFICATE_DIRECTORY=/etc/kubernetes/pki
CA_CERTIFICATE_FILE_PATH=$CA_CERTIFICATE_DIRECTORY/ca.crt
mkdir -p $CA_CERTIFICATE_DIRECTORY
aws eks wait cluster-active            --region=${AWS_DEFAULT_REGION}            --name=${CLUSTER_NAME}
aws eks describe-cluster  --region=${AWS_DEFAULT_REGION}  --name=${CLUSTER_NAME}  --output=text  --query 'cluster.{certificateAuthorityData: certificateAuthority.data, endpoint: endpoint, serviceIpv4Cidr: kuberne
tesNetworkConfig.serviceIpv4Cidr, serviceIpv6Cidr: kubernetesNetworkConfig.serviceIpv6Cidr, clusterIpFamily: kubernetesNetworkConfig.ipFamily, outpostArn: outpostConfig.outpostArns[0], id: id}' > $DESCRIBE_CLUSTE
R_RESULT || rc=$?
B64_CLUSTER_CA=$(cat $DESCRIBE_CLUSTER_RESULT | awk '{print $1}')
APISERVER_ENDPOINT=$(cat $DESCRIBE_CLUSTER_RESULT | awk '{print $3}')
CLUSTER_ID_IN_DESCRIBE_CLUSTER_RESULT=$(cat $DESCRIBE_CLUSTER_RESULT | awk '{print $4}')
OUTPOST_ARN=$(cat $DESCRIBE_CLUSTER_RESULT | awk '{print $5}')
SERVICE_IPV4_CIDR=$(cat $DESCRIBE_CLUSTER_RESULT | awk '{print $6}')
SERVICE_IPV6_CIDR=$(cat $DESCRIBE_CLUSTER_RESULT | awk '{print $7}')
echo $B64_CLUSTER_CA | base64 -d > $CA_CERTIFICATE_FILE_PATH
sed -i s,MASTER_ENDPOINT,$APISERVER_ENDPOINT,g /var/lib/kubelet/kubeconfig
sed -i s,AWS_REGION,$AWS_DEFAULT_REGION,g /var/lib/kubelet/kubeconfig
DOMAIN_NAME=$(echo "$APISERVER_ENDPOINT" | awk -F/ '{print $3}' | awk -F: '{print $1}')
getent hosts "$DOMAIN_NAME" | shuf >> /etc/hosts
sed -i s,CLUSTER_NAME,$CLUSTER_ID,g /var/lib/kubelet/kubeconfig
KUBELET_EXTRA_ARGS="--bootstrap-kubeconfig /var/lib/kubelet/bootstrap-kubeconfig $KUBELET_EXTRA_ARGS"
### kubelet.service configuration
MAC=$(get_meta_data 'latest/meta-data/network/interfaces/macs/' | head -n 1 | sed 's/\/$//')
DNS_CLUSTER_IP="${DNS_CLUSTER_IP}"
KUBELET_CONFIG=/etc/kubernetes/kubelet/kubelet-config.json
echo "$(jq ".clusterDNS=[\"$DNS_CLUSTER_IP\"]" $KUBELET_CONFIG)" > $KUBELET_CONFIG
MAX_PODS_FILE="/etc/eks/eni-max-pods.txt"
set +o pipefail
MAX_PODS=$(cat $MAX_PODS_FILE | awk "/^${INSTANCE_TYPE:-unset}/"' { print $2 }')
MAX_PODS=$(/etc/eks/max-pods-calculator.sh --instance-type-from-imds --cni-version 1.10.0 --show-max-allowed)
echo "$(jq '. += {"evictionHard": {"memory.available": "100Mi", "nodefs.available": "10%", "nodefs.inodesFree": "5%"}}' $KUBELET_CONFIG)" > $KUBELET_CONFIG
echo "$(jq --arg mebibytes_to_reserve "${mebibytes_to_reserve}Mi" --arg cpu_millicores_to_reserve "${cpu_millicores_to_reserve}m"     '. += {kubeReserved: {"cpu": $cpu_millicores_to_reserve, "ephemeral-storage":
"1Gi", "memory": $mebibytes_to_reserve}}' $KUBELET_CONFIG)" > $KUBELET_CONFIG
echo "$(jq ".maxPods=$MAX_PODS" $KUBELET_CONFIG)" > $KUBELET_CONFIG
mkdir -p /etc/systemd/system/kubelet.service.d
cat <<EOF > /etc/systemd/system/kubelet.service.d/10-kubelet-args.conf
cat <<EOF > /etc/systemd/system/containerd.service.d/10-compat-symlink.conf
mkdir -p /etc/docker
bash -c "/sbin/iptables-save > /etc/sysconfig/iptables"
cp -v /etc/eks/iptables-restore.service /etc/systemd/system/iptables-restore.service
sudo chown root:root /etc/systemd/system/iptables-restore.service
systemctl daemon-reload
systemctl enable iptables-restore
echo "$DOCKER_CONFIG_JSON" > /etc/docker/daemon.json
echo "$(jq '.bridge="docker0" | ."live-restore"=false' /etc/docker/daemon.json)" > /etc/docker/daemon.json
systemctl enable kubelet
systemctl start kubelet
```

Прототипирование. Как у вендоров?



Digital ocean

```
root@pool-ilfm5oomo-7pgeo:/var/lib/cloud/scripts/per-instance# find /var/lib/cloud/ -name *k8s*
/var/lib/cloud/scripts/per-instance/000-k8saas
```

```
K8SAAS_PUBLIC_ADDRESS="$(curl --retry 30 -fSs http://169.254.169.254/metadata/v1/interfaces/public/0/ipv4/address)"
K8SAAS_PRIVATE_ADDRESS="$(curl --retry 30 -fSs http://169.254.169.254/metadata/v1/interfaces/private/0/ipv4/address)"
export K8SAAS_PUBLIC_ADDRESS
export K8SAAS_PRIVATE_ADDRESS
K8SAAS_PROVIDER_ID="digitalocean://$(curl --retry 30 -sSf http://169.254.169.254/metadata/v1/id)"
export K8SAAS_PROVIDER_ID
done < <(grep ^k8saas_ < /var/lib/cloud/instance/user-data.txt)

case "${K8SAAS_ROLE:-}" in
  master)
    rm -f /etc/systemd/system/kubelet.service.d/10-kubeadm.conf
    rm -f /lib/systemd/system/kubelet.service
    /opt/k8saas/bootstrap-master
  ;;
  kubelet)
    rm -f /etc/systemd/system/kubelet.service.d/10-kubeadm.conf
    rm -f /lib/systemd/system/kubelet.service
    /opt/k8saas/bootstrap-kubelet
  ;;
  '')
    echo 'Assuming kubeadm usage'
    exit 0
  ;;
  **)
    echo "Don't know role ${K8SAAS_ROLE}. Valid options: [master/kubelet]"
    exit 1
  ;;
esac
~
```

Прототипирование. Как у вендоров?



Digital ocean

Метаданные в файле

```
root@pool-ilfm5oomo-7pgeo:/var/lib/cloud/scripts/per-instance# cat /var/lib/cloud/instance/user-data.txt
#cloud-config
k8saas_role: kubelet
k8saas_master_domain_name: "af639391-5146-463e-af9b-8046228cab32.k8s.ondigitalocean.com"
k8saas_bootstrap_token: "07pgeo.9e081a84bdfc81a9"
k8saas_proxy_token: "a1f8c43e4d63c4e966595b6ae9d4ca12efd40d1574c0e82978838ca986e79334"
k8saas_ca_cert: "-----BEGIN CERTIFICATE-----\nMIIDJzCCAg+gAwIBAgICBnUwDQYJKoZIhvcNAQELBQAwMzEVMBMGA1UEChMMRGln\naXRhbE9jZWFuMRowGAYDVQQDExFrOHNhYXMgQ2x1c3Rl
ciBDQTAeFw0yMjEwMDUx\nMjEyMDNaFw00MjEwMDUxMjEyMDNaMDMxFTATBgNVBAoTDERpZ2l0YWxPY2VhbjEa\nMBgGA1UEAxMRazhzYWFzIENsdXN0ZXIgQ0EwggEiMA0GCSqGSIb3DQEBAQUAA4IB\nDw
AwggEKAoIBAQDCL8aUCVPTXaS40t9DUwEqwgDaksMJyhY9UUXnGNDxPmaeOrY9\nNTLmIylZi32eJ+ZPqTx65RpYznPH1ckqKm4T44f3EBlJ11z5L0rkrJmHWjRWaIi8\nexA25JYRebsGeYg7EEJw2RRfKL
16W6Xb2Tj7JFKsDFB+/GRdlhT7y33T3n+pDGfj\nqaK1iqMTUYPZ9fRn04UyBp+vD8VSLuHXcrCDqojFGR1rJetrPGzmqLQPcmFD6NIS\nI2wyk/morkYHgtEkwU/Yf7k2QcHNTN40k9JHCSz18xKrYDdinl
Cy5VCrT30hNYtP\nd2D9fNvK9gmYgsopMtzLGHzH+Yx0sL1N7hcXAgMBAAGjRTBDMA4GA1UdDwEB/wQE\nAwIBhjASBgNVHRMBAf8ECDAGAQH/AgEAMB0GA1UdDgQWBBTgd5zMf21T3usS2gHC\nWucm5AmY
WjANBgkqhkiG9w0BAQsFAAOCAQEAgoVCgXHDI+Wl+kjY+Hx/Yw8Pnkpm\nWmuJ1XVSnKHcrevr6KCbdGRk9kv7silx0kubfSi7oATwdl08yvK66DBIQTALyQ+J\nH9QSac4f4HaqELwNIeV7/SPDEHlMjPrD
D3LVaiO+KLYPqUrt8+9RlnZx9a5eMn7s\n81DEE5jTFe01aPtZ4r4/kNojiN1xXkXzzRPchrkdFC861/jaFp/ZeT2UvAMT+qbc\ndTcJVaFhz1Dz4fwDZNDZLPepWU2M9KIAUUlu5pcpT99oiWXzlYXwGg4T
gfjmvDxL\nuFz3JpAgA87AzoY93G9NcIMRpI0FekEqFmVqlNJsIyFoxJnA01H38xJ/CA==\n-----END CERTIFICATE-----\n"
k8saas_overlay_subnet: "10.244.0.0/16"
k8saas_cluster_uuid: "af639391-5146-463e-af9b-8046228cab32"
k8saas_dns_service_ip: "10.245.0.10"
k8saas_cp_bridge_ip: "100.65.1.152"
k8saas_node_labels: "doks.digitalocean.com/node-id=77057064-2566-40f8-8b2d-6fbd81591154,doks.digitalocean.com/node-pool-id=9d08454a-74bf-4e3c-a90a-f66086f3a
5f2,doks.digitalocean.com/node-pool=pool-ilfm5oomo,doks.digitalocean.com/version=1.24.4-do.0"
```

Прототипирование. Как у вендоров?



А что это за сервисы такие красивые в GKE

```
gke-my-first-cluster-1-default-pool-3a2323fe-f5m5 ~ # grep -r kube  /var/log/cloud-init-output.log
Created symlink /etc/systemd/system/kubernetes.target.wants/kube-node-installation.service → /etc/systemd/system/kube-node-installation.service.
Created symlink /etc/systemd/system/kubernetes.target.wants/kube-node-configuration.service → /etc/systemd/system/kube-node-configuration.service.
Created symlink /etc/systemd/system/kubernetes.target.wants/kube-container-runtime-monitor.service → /etc/systemd/system/kube-container-runtime-monitor.service.
Created symlink /etc/systemd/system/kubernetes.target.wants/kubelet-monitor.service → /etc/systemd/system/kubelet-monitor.service.
Created symlink /etc/systemd/system/kubernetes.target.wants/kube-logrotate.timer → /etc/systemd/system/kube-logrotate.timer.
Created symlink /etc/systemd/system/kubernetes.target.wants/kube-logrotate.service → /etc/systemd/system/kube-logrotate.service.
Created symlink /etc/systemd/system/multi-user.target.wants/kubernetes.target → /etc/systemd/system/kubernetes.target.
```



```
gke-my-first-cluster-1-default-pool-3a2323fe-f5m5 ~ # systemctl list-dependencies  | grep kube
● ├─kubernetes.target
● │ ├─kube-container-runtime-monitor.service
○ │ ├─kube-logrotate.service
● │ ├─kube-logrotate.timer
● │ ├─kube-node-configuration.service
● │ ├─kube-node-installation.service
● │ └─kubelet-monitor.service
```

Прототипирование. Как у вендоров?



GKE — главное

```
gke-my-first-cluster-1-default-pool-3a2323fe-f5m5 ~ # cat  /home/kubernetes/bin/configure-helper.sh  | wc -l
3725
gke-my-first-cluster-1-default-pool-3a2323fe-f5m5 ~ # cat /home/kubernetes/bin/configure.sh | wc -l
1269
```



Метаданные всё так же забираются со своего API

```
function download-kube-master-certs {
  # Fetch kube-master-certs from GCE metadata server.
  (
    umask 077
    local -r tmp_kube_master_certs="/tmp/kube-master-certs.yaml"
    # shellcheck disable=SC2086
    retry-forever 10 curl ${CURL_FLAGS} \
      -H "X-Google-Metadata-Request: True" \
      -o "${tmp_kube_master_certs}" \
      http://metadata.google.internal/computeMetadata/v1/instance/attributes/kube-master-certs
    # Convert the yaml format file into a shell-style file.
    eval "$(python3 -c '''
```

Прототипирование. Как у вендоров?





Мы уже крутые? Не совсем



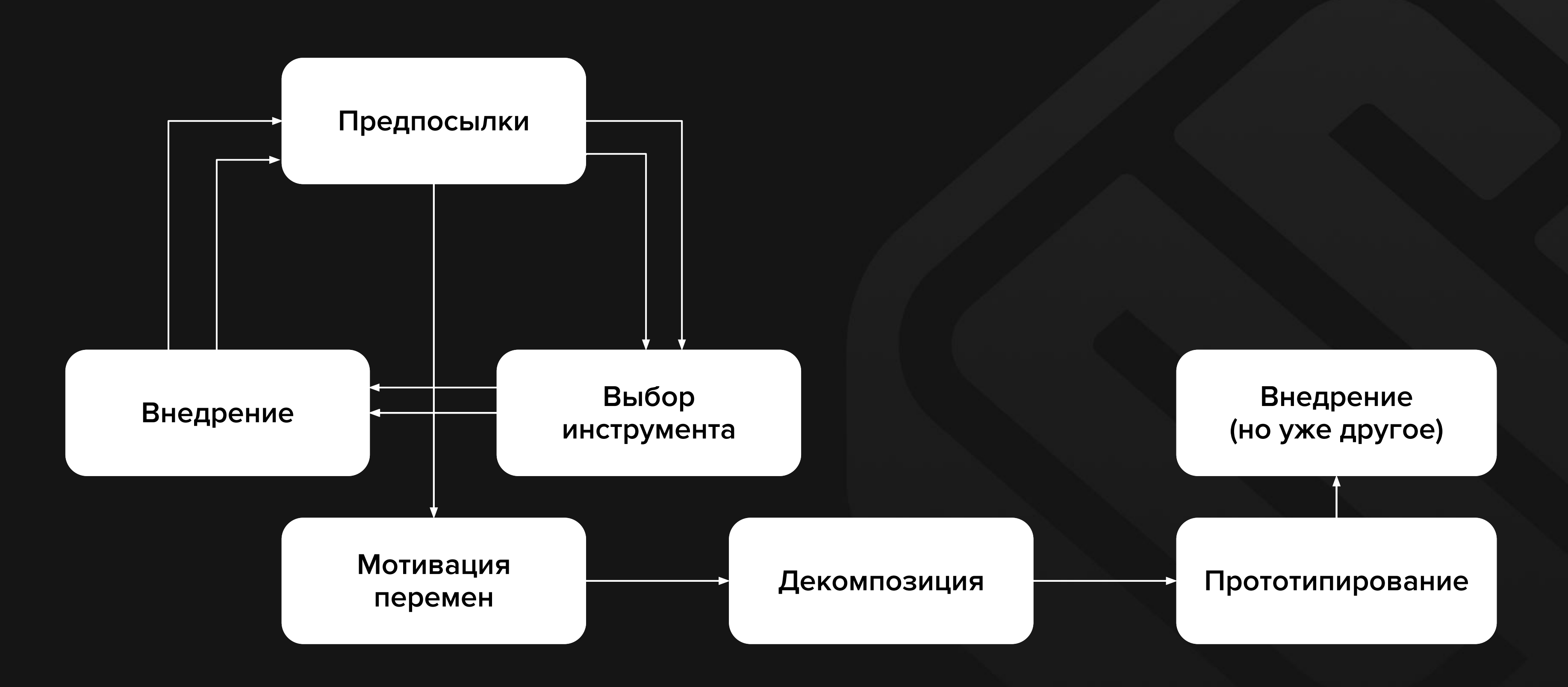
Что имеем:

- HW-кластера с их проблемами всё так же на месте.
- У нас появилась ещё одна точка поддержки.

Какие сделали выводы:

- По-другому можно.
- По-другому несложно.
- Возможно проще, чем было раньше.

Мы уже крутые? Не совсем



Внедрение. Чего минимально не хватает каждой системе



Hardware:

- Система инвентаризации.
- Сервер метаданных.
- Пулл-система обработки метаданных.

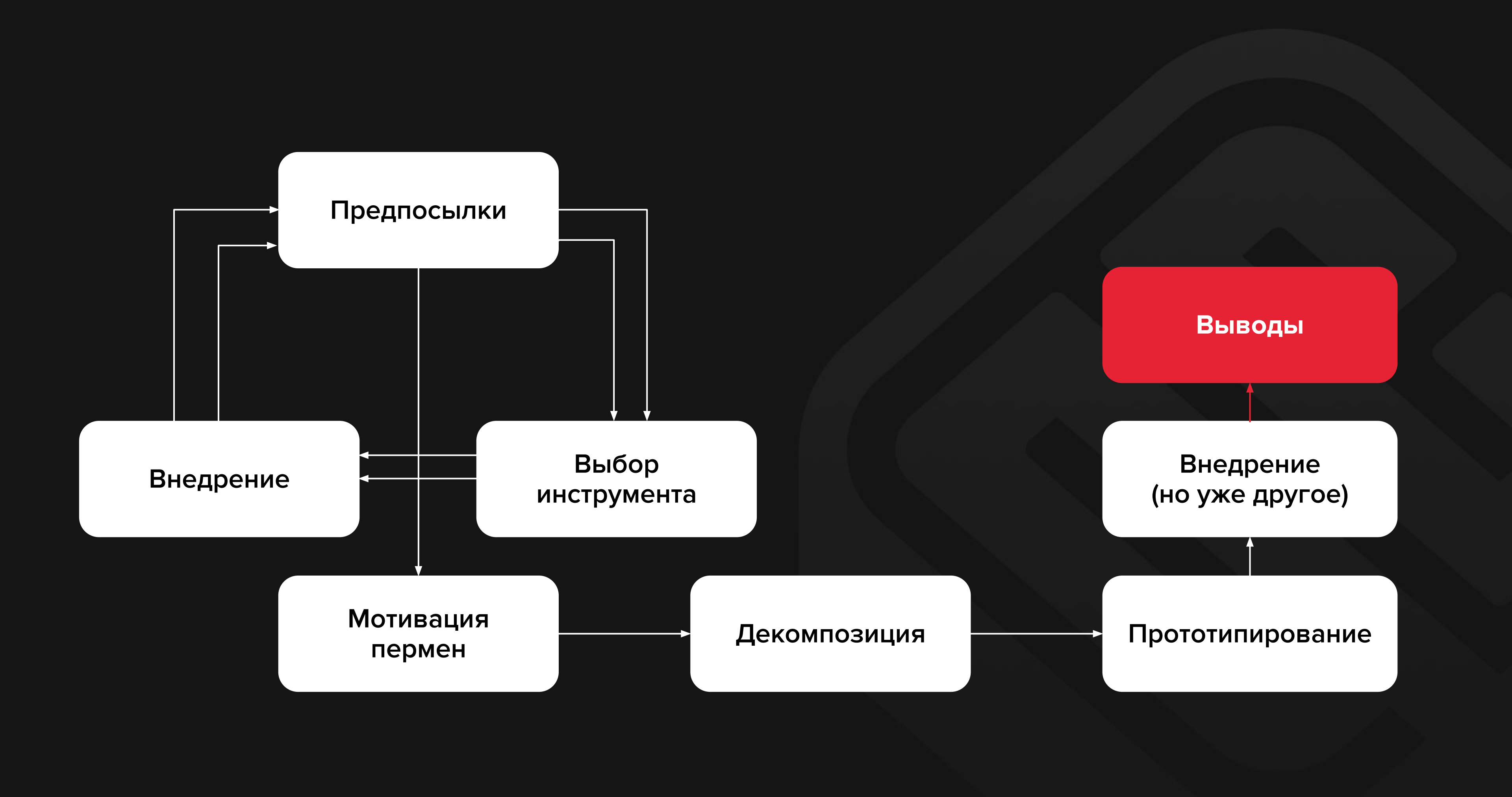
Для облака:

- Процесс скейлинга кластера.
- Процесс апгрейда существующих кластеров.

Внедрить можно за месяц, согласовывать нужно 2 года.



Внедрить несложно, сложно дождаться, когда станет нужно для бизнеса.


Предпосылки
Внедрение
Выбор инструмента
Мотивация пермен
Декомпозиция
Прототипирование
Внедрение (но уже другое)
Выводы

Выводы. Какие проблемы можно приобрести



- Это уже не пуш-система. Сделанная вручную ошибка скорее всего уже будет подхвачена — это типичная проблема пулл-систем.
- Fail fast — must have.
- Нужно решить вопрос A/B-тестирования конфигурации.
- Система становится сложнее:

Было	**Стало**
Добавил в инвентори — пошёл смотреть в лог.	Несколько независимых систем: • provisioner инстансов; • поставщик метаданных; • поставщик снепшотов; • система, оперирующая нодой на основе метаданных.

Выводы



Над нами сейчас вот такие задачи, и именно таким подходом мы хотим их решить:

- Апгрейднуть ядер для 100 нод за час.
- Уверенность в disaster recovery.
- Все окружения раскатываются одинаково.
- Нет зависимости от размера или количества кластеров.

Выводы



В каком порядке мы будем идти дальше:

- Добавить скейл и апгрейд на EC2-кластера.
- Понять все доп. системы, необходимые вокруг Bare Metal.
- Согласовать с другими командами и бизнесом инструментарий.
- Внедрить.
- Перезагрузить ноды кластера.

Выводы. Оглядываясь назад на 5,5 лет



Thank you!



Обратная связь и комментарии по докладу по ссылке



Илья Устинов

@Crowd292

